Universiteit Leiden

Master Computer Science

Rumour Detection for Dutch Twitter

Name:	Nick van der Linden
Student ID:	s2971976
Date:	June 2, 2023
Specialisation:	Data Science
1st supervisor:	Dr. Peter van der Putten
2nd supervisor:	Dr. Jan N. van Rijn

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The internet has become an intricate and indispensable part of modern life, and along with that comes a potent medium for spreading rumours.

This paper focuses on finding rumours in the Dutch language under the constraint of having limited amounts of labelled materials. First, I have gathered a dataset of Dutch tweets from Twitter. These tweets are related to a selection of trending topics in the Netherlands and have been annotated by volunteers. Next, I have employed self-training on XGBoost, support vector machines and naive Bayes models. In addition, I have also examined the importance of features and the generalizability of the results across hashtags. Lastly, I perform a permutation test in combination with upsampling, on these models, a multilayer perceptron, a gradient based classifier and a random forest. The models showed imbalanced predictions, often leaning heavily towards one of the classes. Content features generally provided similar information as context features for all models. The models were not able to reliably generalize across different domains. In other conditions, XGBoost performed the best, achieving AUC scores of around 0.9. The permutation tests with upsampling result in AUC scores between 0.7 and 0.9 for most models.

Models that use only one feature generally perform best, although no one feature that was consistently more prominent. Though limited annotation and data availability held back the model somewhat, this paper could provide a starting point for future research.

Contents

1 Introduction

Modern society has become strongly connected, and that comes with new ways of communicating, finding information, and sending it into the world. This has enabled more people to share their content and discoveries. However, it has also made it easier to spread misleading information. Competing narratives of events are a common occurrence, and perpetrators of misinformation often use covert tactics to disguise their misinformation as factual information. In other words, the internet has opened up numerous new strategies for propagating rumours.

The danger of this is that rumours spread gradually. They might go unnoticed until some event, like a protest or a news item, prompts many people to search for the term. At that point the damage has already been done, sometimes with severe consequences. An example of this is Dylann Roof, who reported googling "Black on white crimes" and becoming radicalized by misleading information, which he found on a website with questionable or even dangerous ties. He went on to commit a mass shooting, partially fuelled by the misleading information he found [1]. Furthermore, there are no easy rules for handling rumours; if left unchallenged, their impact might only increase. Yet, publicly debunking them may only bring more attention to them.

There has been some research on related subjects, most notably in the areas of rumour detection [2], stance detection [3] and fake news detection [4], as well as filter bubbles, search engine optimization [5] and bias [6]. These subareas of research will expanded on in Section 2.

Rumour detection in languages such as Bengali, Hindi, Chinese and Arabic has received some attention [7] [8] [9]. Dutch, too, has been the subject of some research [10] [11] [12] [13]. English data, however, have been the primary focus of research in this field [14] [15] [16]. English is the most commonly spoken language, but findings made on English data may not translate well into other languages that have different lexicons and grammar rules. One result of the focus on English is the relative lack of resources for other languages. This issue has been raised as an issue in numerous studies on rumour detection in languages other than English [8] [17] [2]. Many well-known, easily accessible datasets are in English, whereas similar resources for other languages are scarce. Therefore, gathering new data is often needed to do research in other languages.

A related topic is the type of features used in the literature. Some research uses content features (like punctuation, or the number of capital letters in a message), but research on context features (like network-based or user-specific features, for example) has been more limited. Research using network-based features has been limited to propagation patterns [18], whereas the use of user-specific features is constrained by differences between social media platforms and privacy concerns [19]. Moreover, information is subject to concept drift [20], which means certain features do not always generalize to different contexts very well.

The primary contribution of this thesis is that we will be developing and utilizing a novel, small-scale Dutch Twitter dataset. The goal is not necessarily to achieve high accuracy scores but to examine how well the models perform under challenging circumstances, namely generalizing over different domains and with limited labelled data. I have gathered tweets from a selection of trending Dutch Twitter trends, extracted features related to the context (e.g., user activity) and content of tweets, and built classifiers to distinguish rumours from non-rumours. I will examine the importance of context features vs content features, the importance of static features vs features that are subject to change over time, and the importance of the features overall, as well as to what extent results based on different hashtags carry over to each other. Part of the data I work with is unlabelled, and I have employed self-training to train the models. The code and the data will be made publicly available.[1] [2]

This thesis is structured as follows; Section 2 provides a brief overview of related research fields. Next, In Section 3, I describe the methods I use for building and training the classifier. Then, in Section 4, the process for gathering, exploring and preprocessing the data and the details of the experiments I conduct are outlined. After that, in Section 5, I will discuss the results, followed by the conclusions based on these findings. Lastly, in Section 6, I will describe the overall conclusions of the research and possible future research ideas.

2 Background

This section will provide a brief background on two fields related to the topic of this research, namely, fake news detection and rumour classification. For both of these fields, I will explain what they refer to and give a broad overview of what subareas of research exist in this field. In addition, I will briefly discuss filter bubbles.

2.1 Fake News Detection

Fake news detection refers to identifying news sources that deliberately spread false information. It is a well-studied area of research, and several surveys exist that give an overview of the state-of-the-art in this field [21] [22] [19] [23].

There are a wide variety of papers on fake news detection using algorithms such as convolutional neural networks (CNNs) [4] [24], long short term memory machines (LSTMs) [24], support vector machines (SVMs) [25] [26] and decision trees, among others.

Some of this research has shown impressive results. Meel et al. [4] used a semi-supervised CNN combined with a process called self-ensembling to achieve a classification accuracy of 97.45% on three datasets of news articles posted on Kaggle. Drif et al. [24] implemented a joint CNN and LSTM architecture on the Liar dataset and the News Articles datasets, achieving accuracy scores of 62.34%

[1] https://github.com/NickVanDerLinden/Rumour-Detection
[2] https://www.openml.org/d/45108

and 72.5%, respectively. Reddy et al. [25] employed a gradient-boosting model with word vector representations on the FakeNewsNet and McIntire datasets and achieved accuracy scores of up to 95.49%.

2.2 Filter Bubbles

Related to fake news detection is the concept of filter bubbles. A filter bubble is a state of intellectual isolation, i.e., all the information in the bubble comes from a biased viewpoint. The impact of filter bubbles has also seen some research [27] [28]. They are often approached from a political angle, e.g., the impact they have on ideology [29]. Note that there are some criticisms of the concept as well [30] [31].

2.3 Rumour Classification

Rumour classification is a field with an extensive background. The difference between fake news and a rumour is that a rumour does not have to be false, necessarily. A rumour is a claim that is spread and unverified, while fake news is while fake news is any misinformation, deliberate or otherwise. There are a few main categories of research on rumours: rumour detection, rumour veracity detection, rumour tracking and stance detection.

There are two types of approaches to feature extraction in the context of rumour classification: context-based and content-based. Content features refer to the contents of the medium. For text data, this could be the prevalence of certain symbols or how many capital letters are in the message. Context features refer to the information surrounding the message, such as the number of reactions to it or how active the author is.

Stance detection refers to identifying the attitude of the author of data regarding the topic of that data. It has been used for fact-checking [32] [33], misinformation detection [34], and rumour veracity detection [3], and thus has some overlap between other sub-fields of rumour classification, as well as fake news detection and stance detection. Besides this, it has been used to understand ideological debates [35] [36] or determine the leanings of media outlets [37] [38] [39] [40].

Rumour veracity detection refers to verifying whether certain rumours are true or false. A similar concept is rumour detection, which refers to identifying whether or not the data itself is a rumour or not. Also related is rumour tracking, which refers to tracking a rumour that is known to be a rumour [41].

Like fake news detection, rumour classification is a field with a large body of work dedicated to it, and various surveys outline the state of the art of rumour classification [19] [42]. It has been approached from numerous angles, such as anomaly detection [43], expectation maximization [8], LSTM [44], recurrent neural networks, [9], BERT [45] [46], conditional random fields [14], clustering [47] [48] and XGBoost [2] [49].

Chen et al. [43] approached rumours as anomalies and used factor analysis of mixed data to detect these anomalies. In 2018, the same researchers adopted a similar approach in combination with "crowd wisdom", using unsupervised RNNs. They achieved an F1 score of 89.16% [9]. Alzanin et al. [8] used semi-supervised expectation maximization on a dataset of Arabic tweets, achieving an F1 score of 80%. Zubiaga et al. [14] employed a conditional random field model on Twitter data, detecting rumours using the tweets and the context learned during the event. They report a 40% improvement in the F1 score compared to the competitive baseline. Gumaei et al. [2] use an XGBoost model on Arabic tweets and achieve a 97.16% accuracy score. Akhtar et al. [44] applied a hierarchical LSTM model to perform stance classification and veracity prediction, achieving an 80% accuracy score on the former. Tian et al. [45] used a framework that used multi-lingual models like BERT and a self-training loop. The model was evaluated on Chinese and English datasets and achieved accuracy scores of up to 96%. Anggrainingsih et al. [46] used a combination of BERT and a multi-layer perceptron to achieve an accuracy of 0.87%.

As mentioned in the introduction, while languages besides English have received some attention [8] [9], much research in this area uses English data [14] [15] [16]. Research on Dutch data has been limited, though some research exists in the field of stance detection [10] [11] and veracity detection [12] [13]. This has motivated us to research Dutch data for rumour detection.

Some research has been done on generalizing the results of a model over multiple datasets in one language [17] [50] [51]. Some research has also looked at datasets in different languages, for example, English and Chinese [45], English, Bengali and Hindi [7], or English and Persian [52]. A relevant concept for this issue is the concept of transfer learning, which means using information stored to solve one problem and applying it to solve a different one [53]. The majority of the research on transfer learning for rumour detection has focused on deep learning methods, with traditional machine learning approaches receiving relatively little attention [50]. We will focus on generalizing a model's results using traditional machine learning approaches instead of deep learning.

There has been relatively little research on rumour detection with limited data. Most work concerning limited or noisy data has been in the fields of transfer learning [45] [7] and weakly supervised learning [54]. Weakly supervised learning is a method of machine learning that involves training on noisy or unreliable data to learn limited high-quality data. Han et al. [54] employ weakly supervised learning to perform "Early Rumour Detection": identifying messages that could lead to rumours, before they gain traction.

Some research has been done that focuses on either content [55] [56] [57] [58] or context features [3] [18]. Baheluyan et al. [56] use a combination of content features and cue words on Twitter data and achieve an accuracy score of 0.78. Ma et al. [55] extract features based on TF-IDF and Word2Vec, and use a combination of classifiers. Dungs et al. predict rumour based on only stance and tweeting time

and achieve f1-scores in the range of 80%. Other research has also combined the two in some way. [46] [14] [59]. Hamidian et al. [59] use a combination of content and context rumours with a model that uses decision trees and achieves an 82% accuracy score. Some have suggested that models that combine context and content features perform better [19]. This combination has been explored relatively little in research, however. This has motivated us to combine context and content features

In summary, we will focus on limited data in the Dutch language, using context and content features, as well as traditional machine learning approaches as opposed to deep learning methods.

3 Methods

This section will describe the methods employed in the experiments. These methods include support vector machines, naive Bayes classifiers, XGBoost, and self-training. Since I am interested in using rumour detection on a limited dataset, I have chosen these methods over more complex deep learning or BERT models, which typically require a large dataset [60].

3.1 Support Vector Machines

A support vector machine, or SVM for short, is a supervised learning technique that can be used for classification, regression, and outlier detection. Given a set of training examples, an SVM learns to place a hyperplane boundary that separates the training example into two categories. If the data is not linearly separable, it achieves this by implicitly mapping the points to a higher-dimensional space [61].

It attempts to draw the boundary such that the distance between the data points representing the classes and the decision boundary is as large as possible for either category. Based on this boundary, an SVM can assign new data points to either of the categories based on the training data [61].

3.2 Naive Bayes

Naive Bayes classifiers are a family of probabilistic classifiers. A naive Bayes classifies the probability of a data point belonging to a class by applying the Bayes Theorem. The values of every feature are assumed to be independent of those of other features, allowing one to calculate the conditional probability of a set of features given the class by calculating the conditional probabilities of features separately [62].

There are different classes of naive Bayes classifiers. The main difference between them is in the prior distribution that the features are assumed to have; naive Bayes, for example, is a naive Bayes classifier that operates under the assumption that they follow a normal distribution. This assumption of the distribution of the features is also called the "event model" of the classifier [62].

3.3 XGBoost

XGBoost, also known as extreme gradient boosting, is an algorithm that builds off an approach to machine learning called "boosting" [63]. The idea of boosting is to create a set of weak learners, and combine those weak learners into one strong learner. For each iteration, a new model is created that builds from the previous model and attempts to mitigate its weaknesses. Boosting is not an algorithm so much as a meta-algorithm; it is an approach that can be implemented with any algorithm.

Gradient boosting is a type of boosting algorithm in which subsequent iterations of the model predict the error of their predecessor [64]. These models are typically decision trees. The idea is to approximate an outcome with a weighted combination of weak learners. To find the best combination of functions, gradient boosting applies gradient descent to tweak the weights associated with each weak learner to minimize loss, hence the name gradient boosting.

XGBoost [65] is an implementation of gradient boosting that differs slightly from other gradient boosting algorithms in some areas: instead of gradient descent, XGBoost uses newton boosting to find the best combination of weak learners. Besides this, XGBoost has an extra randomization parameter and allows for a variable number of terminal nodes in a tree [66].

3.4 Self-Training

Since I work with unlabelled data and am interested in what extent I can successfully train models with limited data, I employ self-training [67]. Self-training is an approach to machine learning where the model learns pseudo-labels for unlabelled data. It works in two steps: The first step is to iteratively predict pseudo-labels for the unlabelled data and add these to the training set if they meet a certainty threshold. Then, once a condition is met, the actual task is performed, using supervised or unsupervised classification. This condition could be, for example, when all the data points have been assigned a pseudo-label, or when a large enough percentage of them do.

Self-training has a few benefits compared to more traditional approaches to machine learning. Firstly, it makes data cleaning, preparation and labelling significantly easier. It can also be a possible approach to prevent overfitting. The trade-off is that the labelling accuracy is not always reliable as traditional approaches. This is especially for pseudo-labels.

4 Data

This section will describe the general makeup of the datasets used for the experiments. This includes the PHEME dataset and a new dataset that was gathered using the Twitter API.

	Rumours	non-rumours
Charlie Hebdo	458 (22.0%)	1,621 (78.0%)
Ferguson	284 (24.8%)	859 (75.2%)
German wings	238 (50.7%)	231 (49.3%)
Ottawa	470 (52.8%)	420 (47.2%)
Sydney	522 (42.8%)	699 (57.2%)

Table 1 Rumour to non-rumour ratios and percentages

I will explain what data I have gathered, how this data was gathered, and how it was annotated. This section also includes an overview of the relevant features, both content and context related.

4.1 PHEME dataset

The first dataset I worked with was the PHEME dataset [68]. The PHEME dataset is a well-known dataset of tweets related to five crisis events, namely, the Charlie Hebdo shooting, the Ferguson shooting, the Germanwings crash, the Ottawa shooting and the Sydney siege. The rate of rumours and non-rumours per dataset is described in Table 1.

4.2 Dutch Twitter dataset

The second dataset consists of tweets from Dutch Twitter. Since there are few datasets in Dutch, I chose to create a new dataset. The tweets pertain to three trending hashtags: *#jinek*, *#vleestaks*, and *#inflatie*.

#Jinek is a hashtag related to a popular talk show. The show involves inviting famous people or experts and talking about certain topics, often times political. The show often produces discussion on social media which are a strong potential source of rumours.

#Vleestaks is a hashtag about the taxation of meat. It is a topic that sparked controversy in the Dutch political landscape. Some political parties have proposed raising taxes on meat, the argument being that doing this would lead to less meat consumption, which in turn would be better for the environment as well as for the general health of the Dutch people [69]. This idea has also been faced with some resistance, however [70].

Finally, *#inflatie* is a hashtag about inflation that became popular after inflation rates in the Netherlands in 2022, which rose much sharper than in previous years [71]. This was influenced by the war in Ukraine, which lead to an increase in oil and gas prices [72].

From these hashtags 225,000 tweets were gathered; 175,780 from *#jinek*, 10,542 from *#vleestaks* and 39,278 from *#inflatie*. I filter out replies, retweets and quote tweets to make sure the final dataset consists of original tweets only, in order to minimize the amounts of duplicates tweets that would appear in the data.

To gather these tweets, I used the pytwitter framework to gather tweets related to the relevant hashtags. Pytwitter is a Python library that allows one to access the

Context Features	Content Features
number of followers	presence of question marks
number of tweets in a lifetime	text length
verified user	presence of exclamation marks
account life	ratio of capital to lowercase letters

Table 2 Examples of context vs content features.

Twitter API, which in turn allows one to query Twitter for tweets with specific characteristics. Using this library, I gather two types of data from the tweet: context data, and content data. Context data that is gathered include whether the posters are verified, how old the account is, and public metrics like following count and tweet count. For sake of privacy, location, and name are not gathered and thus not included in the dataset. Content data includes data about the tweet itself: when it was created, the text of the tweet, whether it is has sensitive content, metrics such as like or reply count, and what language it is in. Table 2 shows some examples of context and content features.

Four volunteering annotators labelled the data. Each annotator was presented with the text of 250 tweets randomly sampled from each of the three hashtags, and was given the instruction to identify whether the text was more likely to be a rumour. One hurdle for this process is the fact that the line between rumour and non-rumour is not always clear, and depends on the working definition of a rumour. This is something that has been explored in previous work [19]. To ensure the annotators use a common framework to identify rumours, a set of instructions was given, outlining our definition of a rumour and a non-rumour. The specific instructions are given in the Appendix A.1. For the purpose of this task, a rumour was defined as any claim that is widely spread (or susceptible to being widely spread) and unverified. A non-rumour, in contrast, would be any claim that is either verified, or an opinion.

The end goal for our annotation process was a set of 1000 texts with labels denoting them as either a rumour, or a non-rumour. I made the choice to not have a tweet be labelled by multiple annotators because with the available annotators and data, labelling more data was deemed more useful. Because of this, I could not use metrics like the inter-rater reliability. Note that some tweets used in the sample to be labelled used text that re-occurred in other tweets, so they end up accounting for a bigger part of the data than just 1,000.

4.3 Features

From these tweets, features related to their context and their content were gathered. To decide on a list of features, I performed a literature review of papers [7] [8] [14] [19] [42] [47] [48] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] in rumour detection, to see what features were often used, and compiled them into a list. This list is shown below in Table 3.

From this list, I ignored those features that did not fit the objective of rumour detection. The unused features are listed in a separate column. Some features,

Fig. 1 The steps i took to gather data and choose features

for example, describe attributes that pertain to all tweets in a topic. This is not relevant for this paper, since we are interested in classifying individual tweets. The image feature was excluded for the simple reason that images have URLs that would already show up in the URL feature. Some features occurred little enough that they were excluded for the sake of simplicity.

I also chose to exclude the gender feature, for several reasons: Firstly, this feature does not apply to accounts that belong to groups, organisations, or meme accounts that are not obviously gendered. Moreover, by classifying tweets based on gender, the model could develop a bias based on gender, which in itself could contribute to gender stereotypes. The process of preparing the dataset is depicted in Figure 1.

Used Features		**Unused Features**	
Number of followers	10	Number of positive words	4
Number of tweets in a lifetime	10	Sentiment Score	4
Presence of question marks	9	URL count	4
Presence of a URL	8	Fraction of tweets containing "?"	4
Verified user	8	Presence of images	3
Account life	8	POS tags	3
Followers-following ratio	7	Average tweet length	3
Presence of exclamation marks	7	Fraction of tweets containing ""	3
Ratio of capital to lowercase letters	6	Fraction of tweets containing ""	3
Is retweet	6	Fraction of tweets containing "#"	3
Presence of hashtags	6	Fraction of tweets containing URLs	3
Number of following	6	Fraction of negative tweets	3
Text length	6	Count of negation words	2
Number of lists added to	5	Gender	2
Presence of emoticons	4	Similarity to source tweet	2
Number of reactions	4	Whether the tweet is a source tweet	2
Mentions	4	Number of likes in lifespan	2
Number of likes	4	Number tweets related to topic	2
Time of posting after account registration	4	Fraction of tweets containing emoticons	2
Account activity	3	Average author age	2
first person pronouns	2	Average author followers	2
NSFW content	2		

Table 3 List of features gathered from the Literature Research. From left to right: Features i did use, how often i encountered them, features i did not use, and often i encountered those.

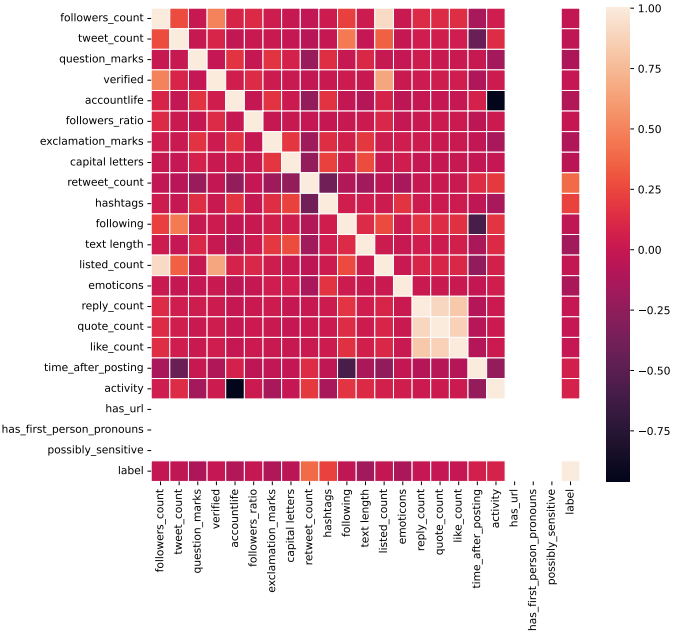

Fig. 2 Correlation matrix on tweets in *#vleestaks*

The final list consists of 22 features. To test the usability of the features, I made a correlation matrix with all of these features. The results indicated that certain features did not correlate with any other feature, including themselves. This happens because there was only one value for these features, throughout the dataset. Based on this result, the $reply\_count$, $quote\_count$, has_url, $has\_first\_person\_pronouns$ and $possibly\_sensitive$ features were dropped. One of the correlation matrices are shown in Figure 2; the correlation matrices with the latest set of features will be added to the Appendix A.

5 Experimental Setup

This section will describe the two experiments conducted to compare the performance of XGBoost, linearSVM and naive Bayes on the data, as well as the permutation test, where additional algorithms are introduced.

The goal of these experiments is to train a model that can distinguish rumour from non-rumours in Dutch, using limited data. I will examine importance of context features vs content features, the importance of static features vs features that are subject to change over time, the importance of the features overall, as well as to what extent results based on one hashtag can be generalized to other hashtags.

5.1 Experiment 1: PHEME Dataset

For the first experiment, I compare the performance of linearSVM, naive Bayes, and XGBoost models on the PHEME dataset. [3]

The performance of these models are compared under two conditions; with and without self-training. In both cases, I perform a train-test split on the data. Then, a percentage of the labels of the training data is hidden. This percentage will be referred to as the mask rate. I experiment with mask rates between 0 and 99 percent. For each mask rate, an ROC curve is computed. The AUC of these ROC curves are plotted as a function of the mask rate.

5.2 Experiment 2: Dutch Twitter Dataset

For the second experiment, the same algorithms are used as in the first experiment. They were trained on the data gathered from Dutch Twitter.

For pre-processing, I scale all features between 1 and 0. This is for two reasons; firstly, some of the features were prone to big differences in value, which scaling between a set range helps mitigate. Secondly, scaling them between 0 and 1 eliminates negative values, which naive Bayes does not take as an input.

I will examine several conditions. The base condition will be a combination of self-training and all the features as described in Section 4.3.

From this condition there are have several axes along which the algorithms will be compared. Firstly, I will conduct experiments without self-training. Unlabelled data in this case would be treated as a third class. This to see to what extent self-training affects the performance of the model.

Besides this, I will also conduct experiments where the data is split by hashtag, training on two of the hashtags and testing on the third. This will give insight as to what extent data from some hashtags carry over to other hashtags.

Additionally, I will conduct experiments with only the "static" features, i.e. features of a tweet that do not change as time goes on. This includes all the features except

[3] https://figshare.com/articles/dataset/PHEME_dataset_of_rumours_and_non-rumours/4010619

for the number of likes, retweets, numbers of lists added to and followers. This experiment aims to provide insight into the relevance of these more time-sensitive features over other, static ones.

I will also examine the influence of context and content features, both with the aforementioned split and without it. This will provide insight into the influence of the content of the data and the context for different hashtags and over all the data as a whole.

Furthermore, I will conduct an experiment where one more feature is added, namely a bag of word representation of the text. This to see if certain specific words could be an indicator for what is a rumour and what is not.

The metric used for evaluation is the AUC score. Additionally, confusion matrices are plotted for each condition.

5.3 Experiment 3: Permutation test

The previous experiment has some potential weaknesses in its design. This includes uneven distribution of labels and hashtags in the data, and possibly duplicates.

For the third experiment, to tackle these potential weaknesses, I sample the data so that every possible combination of label and hashtag occurred the same amount. Furthermore, I filter duplicates from the dataset and scale it in the same way as in the previous experiment.

After pre-processing, I do a permutation test with XGBoost, SVM and naive Bayes, using the labelled part of the dataset, without applying self-training. A permutation test is a procedure where labels of the data are shuffled randomly each timestep, after which a given model is applied to this data. The purpose of this is to test the significance of the relation between the features and the labels, by comparing the performance of our models on the original data with the performance on data with randomized labels.

For every algorithm, I do 100 permutations. Then I apply 5-fold cross-validation on every permutation to split the data into a train and test set. Next, i fit a new model on the train set and monitor its performance on the test set. For all the permutations, the means and standard deviations of the AUC scores are computed. Then, a model is applied on the original data, and the AUC is computed. Lastly, I compute the p-value as the percentage of permutations for which the score obtained is greater than the mean of the scores obtained using the original data.

For the sake of comparison, I apply additional algorithms on this task: RBF SVM, naive Bayes, a random forest, a multi layer perceptron, and a gradient boost classifier.

I also perform an experiment where I add feature selection. I select the x best features before fitting the model to the data for values of x ranging from 1 to 10. The motivation behind this is to filter out features that might add more noise. I

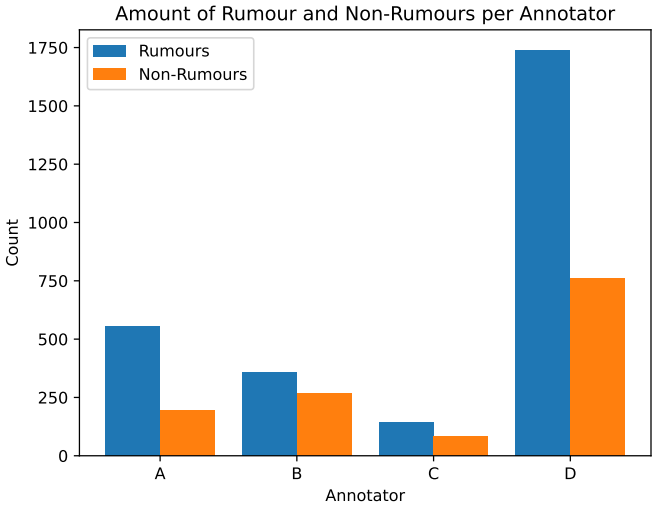

Fig. 3 Amount of Rumours and non-rumours per annotator

ran this experiment with XGBoost, naive Bayes, and SVM, both with a linear and an rbf kernel.

6 Results

This section will describe the results of the conducted experiments. This includes the data analysis, the experiments on the PHEME dataset, and the experiments on the Dutch Twitter dataset under the conditions specified in Section 5.

6.1 Exploratory Data Analysis

This subsection outlines the results of the exploratory data analysis. Figure 3 describes how many tweets are labelled as rumours and non-rumours by each annotator (denoted here as A, B, C, and D). In this case, all the annotators account for more rumours than non-rumours. The annotators reached an amount of annotated tweets well above the 250 they were assigned, which means the tweets in their sample contained doubles. Since the annotated tweets did not include IDs, every tweet that matched the text the annotators had labelled was assigned a label. The re-occurring text could be a sign that of a network of bots tweeting pre-written text.

The main difference between the annotators is the number of rumours their annotations cover. Though, it is difficult to say how much of the difference is because of tweets with duplicate text and how much is due to individual differences.

One interesting fact to note from Figure 4 is that, while for *#inflatie* and *#jinek* the distribution of rumour and non-rumours is equal, the distribution for

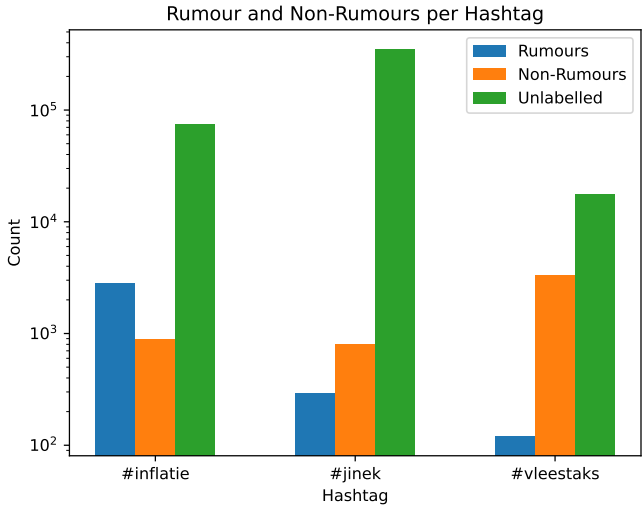

Fig. 4 Rumours and non-rumours per hashtag

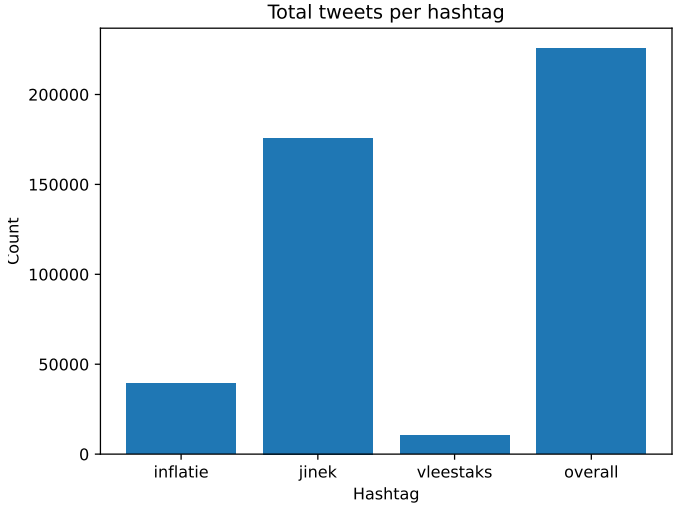

Fig. 5 Amount of Tweets per hashtag

#vleestaks skews toward non-rumours. Another thing to note is that while *#jinek* has more tweets than the other two hashtags, it has the least amount of rumours and non-rumours. This difference is because the latter two hashtags have relatively more re-occurring texts. The annotations end up accounting for more tweets in total as as result of one of these texts being annotated.

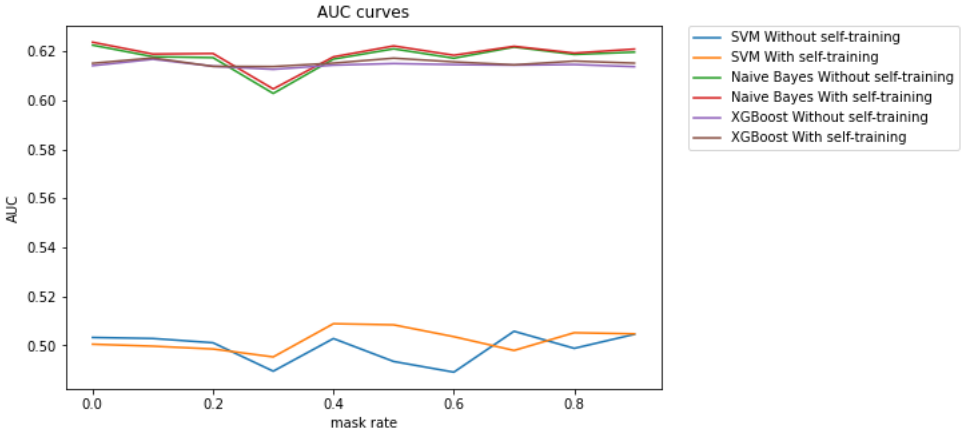

Fig. 6 AUC curve for models with and without self-training on the PHEME dataset for mask rates of the training data from 0 to 0.99. Top left: naive Bayes, top right: linearSVM, bottom: XGBoost

Figure 5 shows that *#jinek* accounts for many more tweets than *#inflatie* and *#vleestaks*. That is likely because the show Jinek (and the corresponding hashtag related to it) has been around for some time. In contrast, the other hashtags related to events that were relatively new at the time of gathering the data.

6.2 Experiment 1: PHEME Dataset

In this subsection, I will briefly show the result of the first experiment.

Figure 6 shows the AUC curves for the different mask rates for XGBoost, naive Bayes and SVM on the PHEME dataset. Overall, the AUC score on this dataset is lower than the AUC curve in the general literature on rumour detection [83] [84] [85]. Though in the literature related to weakly supervised learning in rumour detection specifically, the AUC scores are comparable to our results [23].

The performance of the naive Bayes model initially goes down as the mask rate increases, dropping from 0.62 with a mask rate of 0 to 0.60 with a mask rate of 0.3. The fact that performance does not drop with an increasing mask rate is surprising since one would expect a higher mask rate to make the task more difficult. The highest mask rate used was 0.99, which leaves 1000 labelled examples. This experiment suggests that those 1000 labelled examples are enough to perform well on this task. The addition of self-training does improve the performance a little overall, though the difference is negligible.

The XGBoost model shows relatively consistent performance, consistently scoring around 0.61. This result suggests that XGBoost is better able to handle incomplete data. Self-training makes a small contribution to performance here, accounting for an increase in performance of about 0.001 compared to the condition without self-training.

Model / features	AUC
Majority class classifier (baseline)	0.50
XGBoost/all features	0.95
XGBoost/content features	0.95
XGBoost/context features	0.96
linearSVM/all features	0.52
linearSVM/content features	0.49
linearSVM/context features	0.51
naive Bayes/all features	0.52
naive Bayes/content features	0.61
naive Bayes/context features	0.50

Table 4 Table of experiments and results for XGBoost, linearSVM and naive Bayes with self-training, all features across all hashtags compared to context-only and content-only features

The performance of the linearSVM model fluctuates while not showing any trends in performance changes with increasing mask rates. The AUC score fluctuates around the 0.5 mark, this indicates that linearSVM classifies almost by chance. This is in contrast with XGBoost and naive Bayes, both of which show a weak predictive ability. Since all experiments have been carried out within the same framework, this low performance is somewhat puzzling. Further testing has not provided more insights into the cause of this.

6.3 Experiment 2: Dutch Twitter Dataset

The baseline we use here is the majority classifier. It predicts the majority class every time, based on the number of non-rumours and rumours.

The AUC scores differ slightly. For some, the AUC reaches around 0.5. For a few, it falls below 0.5, which would be worse than random. For some however, it reaches 0.8 or higher.

That could be because of problems with the data. The features might not give enough information to train the model to distinguish between rumours and non-rumours. Another possibility could be that the models are overfitting the training data, causing it to perform worse on the test data. Lastly, the data is unbalanced, which may lead to worse performance overall.

Also, note that SVM and naive Bayes both score lower than on the PHEME dataset. That could partially be because the Dutch Twitter data is unbalanced. However, the PHEME dataset has a similar problem: Holding more non-rumours than rumours. Since the Dutch dataset contains more data than the PHEME dataset, a more likely explanation could be that the data has noise that causes worse performance.

Without self-training, the results are slightly worse than with self-training. This would make sense since the pseudo-labels generated in self-training are less reliable than labels generated from annotations. Despite this, XGBoost achieves high AUC scores for both conditions.

Model / features	AUC
Majority class classifier (baseline)	0.50
XGBoost/all features	0.97
XGBoost/no selftr./all features	0.99
XGBoost/early features	0.96
XGBoost/all features/BOW	0.55
linearSVM/all features	0.52
linearSVM/no selftr./all features	0.83
linearSVM/early features	0.5
linearSVM/all features/BOW	0.52
naive Bayes/all features	0.52
naive Bayes/no selftr./all features	0.60
naive Bayes/early features	0.60
naive Bayes/all features/BOW	0.51

Table 5 Table of experiments and results for XGBoost, linearSVM and naive Bayes, self-training with all features across all hashtags and "early" features, as well as all features without self-training.

Table 4 shows the results for the three algorithms with the complete set of features and with a subset of the features. Overall, the performance is slightly better than random. Naive Bayes performs slightly better than chance level with content features and otherwise around the chance level. LinearSVM performs around chance level all around. Overall, content and context features are equally indicative of rumours. XGBoost achieves a good performance overall, achieving AUC scores near 1. This is surprising, especially in contrast with the other results. Although it is unclear what is causing this, we can rule out leakage since this performance is not as high in other conditions or algorithms.

Table 5 shows the results for the three algorithms, both with all the features and with various subsets of the features. The XGBoost model shows good performance overall, similar to the results shown in Table 4. All the models do relatively well without self-training. For the complete feature set, the AUC is slightly over 0.5, but this seems to be a negligible increase. These results point to some common factor in the dataset, or the chosen features extracted from it, which leads to poor performance. Adding a bag of words representation to the complete list of features does not lead to a significant difference in performance, which indicates that knowing the exact words used does not provide much extra information. In fact, for XGBoost, it leads to notable decrease in performance. One possible explanation is that, since the data are from three different hashtags, they would all have different frequencies for the recurrences of certain words.

Table 6 show the AUC scores of the algorithms on the data, split so that *#inflatie* and *#vleestaks* are the train set and *#jinek* is the test set.

LinearSVM performs worse than the chance level with context-only features but otherwise shows a moderate predictive power. This suggests that content features provide more information than context features about the presence of rumours. This is contradicted by XGBoost & naive Bayes. XGBoost performs best with context-only features but around chance level with content features and worse

with both. Naive Bayes performs worse with content features and around chance level otherwise.

Overall, these results imply that features from the *#inflatie* and *#vleestaks* hashtags do not carry over well to tweets from the *#jinek* hashtag.

The results in Tables 7 and 8 show a mostly chance level performance for the algorithms, one noteworthy exception being XGBoost, which has rather dramatic AUC scores. The cause of this is not clear. They can be partly explained by looking at the confusion matrices in Table 9, which indicate that XGBoost tends towards rumours, but this is likely not the entire reason.

Another thing to note is the low score for linearSVM with context features, which suffers from a similar issue. The high score in Table 7 with content features further suggests that the content features of *#jinek* and *#inflatie* carry over particularly well to *#vleestaks*, as opposed to the context features.

Table 10 shows a good performance for XGBoost. XGBoost achieved an AUC score of 0.8 or higher for this condition, as well as for most of the conditions shown in Table 5. This is in contrast with the results shown in Table 9. This difference is also reflected in the AUC scores in Table 7. The lowest AUC scores in this table are a product of the tendency of the model to predict rumours, in a dataset that holds more non-rumours. This figure indicates that the tweets from *#inflatie* are heavily skewed towards non-rumours, which results in more extreme AUC scores.

Table 11 shows that naive Bayes, for this condition, has a tendency to predict mostly rumours. This is a pattern that re-occurred relatively often, particular for conditions where the AUC was around 0.5 or lower.

Model / features	**AUC**
XGBoost/all features	0.4
XGBoost/context features	0.58
XGBoost/content features	0.53
linearSVM/all features	0.61
linearSVM/context features	0.38
linearSVM/content features	0.56
naive Bayes/all features	0.52
naive Bayes/context features	0.50
naive Bayes/content features	0.40

Table 6 Table of experiments and results for XGBoost, linearSVM and naive Bayes, split so that tweets related to *#jinek* are the test set, and the train set consists of tweets related to *#vleestaks* and *#inflatie*.

Model / features	AUC
XGBoost/all features	0.16
XGBoost/context features	0.006
XGBoost/content features	0.89
linearSVM/all features	0.50
linearSVM/context features	0.10
linearSVM/content features	0.50
naive Bayes/all features	0.50
naive Bayes/context features	0.50
naive Bayes/content features	0.50

Table 7 Table of experiments and results for XGBoost, linearSVM and naive Bayes, split so that tweets related to *#vleestaks* are the test set, and the train set consists of tweets related to *#jinek* and *#inflatie*.

Model / features	AUC
XGBoost/all features	0.46
XGBoost/context features	0.44
XGBoost/content features	0.5
linearSVM/all features	0.42
linearSVM/context features	0.50
linearSVM/content features	0.50
naive Bayes/all features	0.50
naive Bayes/context features	0.50
naive Bayes/content features	0.50

Table 8 Table of experiments and results for XGBoost, linearSVM and naive Bayes, split so that tweets related to *#inflatie* are the test set, and the train set consists of tweets related to *#vleestaks* and *#jinek*.

		Prediction label	
		Non-Rumour	**Rumour**
Actual Label	**Non-Rumour**	20	1600
	Rumour	60	0

Table 9 Confusion Matrix for XGBoost, using only context features and with the data split so that tweets related to *#vleestaks* are the test set, and the train set consists of tweets related to *#inflatie* and *#jinek*.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	1300	340
	Rumour	0	60

Table 10 Confusion Matrix for XGBoost, using only content features and with the data split so that tweets related to *#vleestaks* are the test set, and the train set consists of tweets related to *#inflatie* and *#jinek*.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	19	490
	Rumour	0	310

Table 11 Confusion matrix for naive Bayes, using all features and all the hashtags, as well as self-training.

Method	AUC	p-value
XGBoost	0.81	0.01
MNB	0.61	0.01
LSVM	0.70	0.01
RF	0.79	0.01
MLP	0.78	0.01
GBC	0.81	0.01
SVM	0.78	0.01
GNB	0.52	0.01

Table 12 Results of the experiments with upsampling for each method tested. From left to right: the AUC scores and the p-value.

6.4 Experiment 3: Permutation test

Table 12 shows that the general performance is much better than the performance of these same algorithms in the second set of experiments. This difference can be explained by the introduction of the pre-processing steps described earlier, namely up-sampling and scaling.

The algorithms that show the most predictive power are XGBoost, random forest and gradient boost classifier, all reaching AUC scores around 0.8. Multilayer perceptron and support vector machines follow, with performances in the 0.7s. Lastly, gaussian and multinomial naive Bayes achieve AUC scores in the range of 0.5-0.6, barely above the chance level. The relatively poor performance of both of the naive Bayes models could indicate that the underlying assumptions of the distribution of the data are not being met, causing the model to have a relatively poor fit in this particular task.

Figure 7 shows that there is no single feature that really jumps out for XGBoost, Random Forest, the Gradient Boost Classifier. LinearSVM shows a different pattern from the other algorithms, with the time of posting having a strong positive feature importance, and activity, text length, retweet count and tweet count showing a strong negative importance.

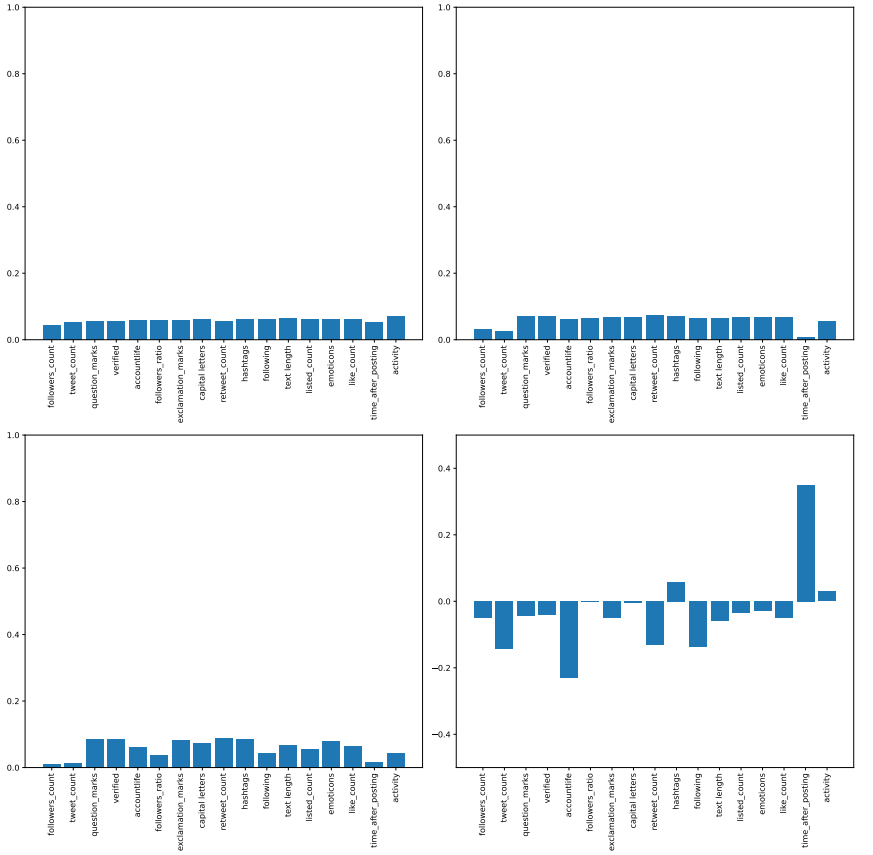

Fig. 7 Feature importances, from left to right, top to bottom: XGBoost, Random Forest, Gradient Boost Classifier, linearSVM

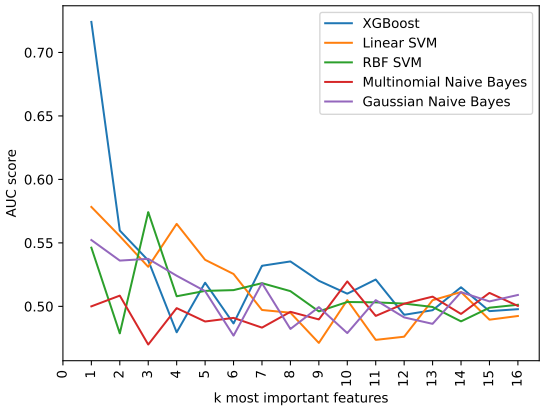

Fig. 8 AUC scores for top-k feature selection for the XGBoost model, averaged over ten runs. From left to right, top to bottom: XGBoost, multinomial naive Bayes, linearSVM, RBF SVM, gaussian naive Bayes

Figure 8 shows that for most algorithms, using only one feature leads to the best results, with the performance decreasing to chance level afterwards. The exception is RBF SVM, where there is another peak at k = 3. It seems that while the most important feature is a useful predictor, adding other features adds information that is in some way conflicting, and leads to worse results. This could be an artefact of how the annotators labelled the data. Another explanation could be that the set of features do not adequately capture the characteristics of a rumour. With the previous plot in mind, which feature would be the most important here is unclear.

7 Discussion

7.1 Findings

On the Dutch data, XGBoost outperformed the other models, with AUC scores consistently around 0.9, outside of the split conditions. The other models, by contrast, achieved scores of between 0.5 and 0.6 for these conditions. A re-occurring pattern was that models would often classify most data as one class over the other. This was likely caused by the somewhat unbalanced data.

The performance was not aided by including the bag of words feature. For the model using only the static features, the performance did not significantly improve or worsen compared to the model using all the features.

Experiments with content features did not show significantly better results than experiments with context features. The results of the experiments on generalizing to other hashtags were inconsistent; in terms of AUC, models either performed around chance level, notably poorly or notably well, This indicates that this manner of generalizing was not successful.

In the experiment described in Section 6.4, XGBoost and SVM obtained better performance. Performance for both gaussian and multinomial naive Bayes remained relatively poor. A likely reason for this is that the assumptions about the data did not hold.

7.2 Limitations

One issue I encountered while gathering data is that many tweets about events in the Netherlands are in English, making it harder to collect tweets in Dutch.

Another issue is that we worked with relatively few labellers, preventing us from examining the inter-annotator agreement in our research and affecting the reliability of the labels. This also lead to some balancing issues, affecting the performance in the experiments on the Dutch Twitter Dataset.

Furthermore, rumours spread on more platforms than Twitter; this paper does not examine other social media platforms like Facebook or Tiktok, let alone real life. Depending on the platform, rumours could spread differently, and different rumours might be more prevalent than others on different platforms.

Lastly, our research does not into go what the best way is to handle rumours. That includes questions like which ones harmful, or at what point it is better to ignore them or to step in.

7.3 Future work

The thesis can hopefully prove to be a starting point for future work. One suggestion could be to employ more annotators to label data and employ an inter-annotator agreement metric to ensure quality labels. Moreover, one could examine

different sources, like Reddit or Instagram. Lastly, A type of cost-sensitive classification or experimenting with different thresholds on model scores could mitigate the low AUC scores of the models.

8 Conclusion

In this thesis, my goal was to build a model that would be able to identify rumours in the Dutch language. I have created models that could identify rumours in tweets gathered from three different Twitter hashtags about topics trending in the Netherlands, with an accuracy of around 0.6. The models tended to classify all tweets as one class. Content features generally provided slightly more information than context features for all models. The permutation test experiments were successful, with performance falling between 0.7 and 0.9. The length of the text was the strongest predictor, and the best-performing models used just the text length.

References

[1] P. H. L. Bernstein, S. Horwitz, "Dylann Roof's racist manifesto: 'I have no choice'," *The Washington Post*, 06 2015.

[2] A. Gumaei, M. S. Al-Rakhami, M. M. Hassan, V. H. C. De Albuquerque, and D. Camacho, "An effective approach for rumour detection of Arabic tweets using extreme gradient boosting method," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 21, no. 1, pp. 1–16, 2022.

[3] S. Dungs, A. Aker, N. Fuhr, and K. Bontcheva, "Can rumour stance alone predict veracity?," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3360–3370, 2018.

[4] P. Meel and D. K. Vishwakarma, "A temporal ensembling based semi-supervised convnet for the detection of fake news articles," *Expert Systems with Applications*, vol. 177, p. 115002, 2021.

[5] D. Sharma, R. Shukla, A. K. Giri, and S. Kumar, "A brief review on search engine optimization," in *2019 9th International Conference on Cloud Computing, Data Science Engineering*, pp. 687–692, 2019.

[6] B. Fortuna, C. Galleguillos, and N. Cristianini, "Detection of bias in media outlets with statistical learning methods," in *Text Mining*, pp. 57–80, Chapman and Hall/CRC, 2009.

[7] D. Kar, M. Bhardwaj, S. Samanta, and A. P. Azad, "No rumours please! A multi-indic-lingual approach for COVID fake-tweet detection," in *2021 Grace Hopper Celebration India (GHCI)*, pp. 1–5, IEEE, 2020.

[8] S. M. Alzanin and A. M. Azmi, "Rumour detection in Arabic tweets using semi-supervised and unsupervised expectation–maximization," *Knowledge-Based Systems*, vol. 185, p. 104945, 2019.

[9] W. Chen, Y. Zhang, C. K. Yeo, C. T. Lau, and B. S. Lee, "Unsupervised rumour detection based on users' behaviours using neural networks," *Pattern Recognition Letters*, vol. 105, pp. 226–233, 2018.

[10] S. Wang, M. Schraagen, E. T. K. Sang, and M. Dastani, "Public sentiment on governmental COVID-19 measures in dutch social media," in *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, 2020.

[11] N. Bauwelinck and E. Lefever, "Annotating topics, stance, argumentativeness and claims in Dutch social media comments: A pilot study," in *Proceedings of the 7th workshop on argument mining*, pp. 8–18, 2020.

[12] B. Verhoeven and W. Daelemans, "Clips stylometry investigation (csi) corpus: A Dutch corpus for the detection of age, gender, personality, sentiment and deception in text," in *LREC 2014-Ninth international conference on language research and evaluation*, pp. 3081–3085, 2014.

[13] B. Berendt, P. Burger, R. Hautekiet, J. Jagers, A. Pleijter, and P. Van Aelst, "Factrank: Developing automated claim detection for Dutch-language fact-checkers," *Online Social Networks and Media*, vol. 22, p. 100113, 2021.

[14] A. Zubiaga, M. Liakata, and R. Procter, "Learning reporting dynamics during breaking news for rumour detection in social media," *arXiv preprint arXiv:1610.07363*, 2016.

[15] E. Kochkina, M. Liakata, and I. Augenstein, "Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm," *arXiv preprint arXiv:1704.07221*, 2017.

[16] J. Ma, W. Gao, and K.-F. Wong, "Rumour detection on Twitter with tree-structured recursive neural networks," Association for Computational Linguistics, 2018.

[17] M. Lukasik, K. Bontcheva, T. Cohn, A. Zubiaga, M. Liakata, and R. Procter, "Gaussian processes for rumour stance classification in social media," *ACM Transactions on Information Systems*, vol. 37, no. 2, pp. 1–24, 2019.

[18] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning," Association for Computational Linguistics, 2017.

[19] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Information Sciences*, vol. 497, pp. 38–55, 2019.

[20] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.

[21] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[22] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–40, 2020.

[23] K. Sharma, F. Qian, H. Jiang, N. Ruchansky, M. Zhang, and Y. Liu, "Combating fake news: A survey on identification and mitigation techniques," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 3, pp. 1–42, 2019.

[24] A. Drif, Z. F. Hamida, and S. Giordano, "Fake news detection method based on text-features," in *The Ninth International Conference on Advances in Information Mining and Management*, 2019.

[25] H. Reddy, N. Raj, M. Gala, and A. Basava, "Text-mining-based fake news detection using ensemble methods," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 210–221, 2020.

[26] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *International conference on intelligent, secure, and dependable systems in distributed and cloud environments*, pp. 127–138, Springer, 2017.

[27] S. Flaxman, S. Goel, and J. M. Rao, "Filter bubbles, echo chambers, and online news consumption," *Public opinion quarterly*, vol. 80, no. S1, pp. 298–320, 2016.

[28] D. Spohr, "Fake news and ideological polarization: Filter bubbles and selective exposure on social media," *Business Information Review*, vol. 34, no. 3, pp. 150–160, 2017.

[29] E. Pariser, *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin, 2011.

[30] P. M. Dahlgren, "A critical review of filter bubbles and a comparison with selective exposure," *Nordicom Review*, vol. 42, no. 1, pp. 15–33, 2021.

[31] A. Bruns, *Are filter bubbles real?* John Wiley & Sons, 2019.

[32] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: A large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.

[33] A. Hanselowski, C. Stab, C. Schulz, Z. Li, and I. Gurevych, "A richly annotated corpus for different tasks in automated fact-checking," *arXiv preprint arXiv:1911.01214*, 2019.

[34] Y. Hou, P. van der Putten, and S. Verberne, "The covmis-stance dataset: Stance detection on Twitter for COVID-19 misinformation," *arXiv preprint arXiv:2204.02000*, 2022.

[35] I. Habernal, H. Wachsmuth, I. Gurevych, and B. Stein, "The argument reasoning comprehension task: Identification and reconstruction of implicit warrants," *arXiv preprint arXiv:1708.01425*, 2017.

[36] J. Vamvas and R. Sennrich, "X-stance: A multilingual multi-target dataset for stance detection," *arXiv preprint arXiv:2003.08385*, 2020.

[37] P. Stefanov, K. Darwish, A. Atanasov, and P. Nakov, "Predicting the topical stance and political leaning of media using tweets," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 527–537, 2020.

[38] S. Mukherjee and G. Weikum, "Leveraging joint interactions for credibility analysis in news communities," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 353–362, 2015.

[39] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum, "Where the truth lies: Explaining the credibility of emerging claims on the web and social media," in *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 1003–1012, 2017.

[40] M. Hardalov, A. Arora, P. Nakov, and I. Augenstein, "A survey on stance detection for mis-and disinformation identification," *arXiv preprint arXiv:2103.00242*, 2021.

[41] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Computing Surveys*, vol. 51, no. 2, pp. 1–36, 2018.

[42] D. Varshney and D. K. Vishwakarma, "A review on rumour prediction and veracity assessment in online social network," *Expert Systems with Applications*, vol. 168, p. 114208, 2021.

[43] W. Chen, C. K. Yeo, C. T. Lau, and B. S. Lee, "Behaviour deviation: An anomaly detection view of rumour preemption," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference*, pp. 1–7, IEEE, 2016.

[44] M. S. Akhtar, A. Ekbal, S. Narayan, and V. Singh, "No, that never happened!! investigating rumours on Twitter," *IEEE Intelligent Systems*, vol. 33, no. 5, pp. 8–15, 2018.

[45] L. Tian, X. Zhang, and J. H. Lau, "Rumour detection via zero-shot cross-lingual transfer learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 603–618, Springer, 2021.

[46] R. Anggrainingsih, G. M. Hassan, and A. Datta, "BERT based classification system for detecting rumours on Twitter," *arXiv preprint arXiv:2109.02975*, 2021.

[47] C. Raj and P. Meel, "Is dynamic rumour detection on social media viable? an unsupervised perspective," *arXiv preprint arXiv:2111.11982*, 2021.

[48] C. Chang, Y. Zhang, C. Szabo, and Q. Z. Sheng, "Extreme user and political rumour detection on Twitter," in *International conference on advanced data mining and applications*, pp. 751–763, Springer, 2016.

[49] J. Shang, J. Shen, T. Sun, X. Liu, A. Gruenheid, F. Korn, Á. D. Lelkes, C. Yu, and J. Han, "Investigating rumour news using agreement-aware search," in *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 2117–2125, 2018.

[50] R. Sicilia, L. Francini, and P. Soda, "Representation and knowledge transfer for health-related rumour detection," in *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 591–596, IEEE, 2021.

[51] M. Guo, Z. Xu, L. Liu, M. Guo, and Y. Zhang, "An adaptive deep transfer learning model for rumour detection without sufficient identified rumours," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[52] M. Ghayoomi and M. Mousavian, "Deep transfer learning for COVID-19 fake news detection in persian," *Expert Systems*, p. e13008, 2022.

[53] S. Pargaien, D. Singh, R. Prakash, V. P. Dubey, H. Pant, and A. V. Pargaien, "Land use classification of kathgodam region using transfer learning-based approach," in *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*, pp. 1–7, IEEE, 2022.

[54] S. Han, *Context-aware message-level rumour detection with weak supervision*. PhD thesis, University of Sheffield, 2020.

[55] J. Ma and Y. Luo, "The classification of rumour standpoints in online social network based on combinatorial classifiers," *Journal of Information Science*, vol. 46, no. 2, pp. 191–204, 2020.

[56] H. Bahuleyan and O. Vechtomova, "Uwaterloo at SemEval-2017 task 8: Detecting stance towards rumours with topic independent features," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 461–464, 2017.

[57] Y. Qin, D. Wurzer, V. Lavrenko, and C. Tang, "Spotting rumours via novelty detection," *arXiv preprint arXiv:1611.06322*, 2016.

[58] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumours from microblogs with recurrent neural networks," 2016.

[59] S. Hamidian and M. T. Diab, "Rumour detection and classification for Twitter data," *arXiv preprint arXiv:1912.08926*, 2019.

[60] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, vol. 2. Springer, 2009.

[61] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[62] C. Zhai and S. Massung, *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool, 2016.

[63] R. E. Schapire, "A brief introduction to boosting," in *IJCAI*, vol. 99, pp. 1401–1406, 1999.

[64] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, 2021.

[65] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, *et al.*, "XGBoost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[66] D. Nielsen, "Tree boosting with XGBoost-why does XGBoost win "every" machine learning competition?," Master's thesis, NTNU, 2016.

[67] J. Tanha, M. Van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, 2017.

[68] A. Zubiaga, G. W. S. Hoi, M. Liakata, and R. Procter, "PHEME dataset of rumours and non-rumours," 10 2016.

[69] I. Odegard and G. Bergsma, *Milieueffecten van verbeteropties voor de Nederlandse eiwitconsumptie*. CE Delft, 2012.

[70] F. Heukelom and E.-M. Sent, "Verhalen vanuit de gedragseconomie-www-publicatie," 2010.

[71] "Netherlands inflation rate," Nov 2022.

[72] S. Menon, "War and gas: What Russia's war on Ukraine means for energy prices and the climate," *Environmental Defense Fund*, May 2022.

[73] E. W. Pamungkas, V. Basile, and V. Patti, "Stance classification for rumour analysis in Twitter: Exploiting affective information and conversation structure," *arXiv preprint arXiv:1901.01911*, 2019.

[74] I. Baris, L. Schmelzeisen, and S. Staab, "Clearumour at SemEval-2019 task 7: Convolving elmo against rumours," *arXiv preprint arXiv:1904.03084*, 2019.

[75] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," in *Proceedings of the 20th international conference on World wide web*, pp. 675–684, 2011.

[76] V. Qazvinian, E. Rosengren, D. Radev, and Q. Mei, "Rumour has it: Identifying misinformation in microblogs," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, pp. 1589–1599, 2011.

[77] F. Yang, Y. Liu, X. Yu, and M. Yang, "Automatic detection of rumour on sina weibo," in *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pp. 1–7, 2012.

[78] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumours in social media from enquiry posts," in *Proceedings of the 24th international conference on world wide web*, pp. 1395–1405, 2015.

[79] K. Wu, S. Yang, and K. Q. Zhu, "False rumours detection on sina weibo by propagation structures," in *2015 IEEE 31st international conference on data engineering*, pp. 651–662, IEEE, 2015.

[80] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumour propagation in online social media," in *2013 IEEE 13th international conference on data mining*, pp. 1103–1108, IEEE, 2013.

[81] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, "Detect rumours using time series of social context information on microblogging websites," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 1751–1754, 2015.

[82] S. Kwon, M. Cha, and K. Jung, "Rumour detection over varying time windows," *PloS one*, vol. 12, no. 1, p. e0168344, 2017.

[83] A. Kumar, M. Bhatia, and S. R. Sangwan, "Rumour detection using deep learning and filter-wrapper feature selection in benchmark Twitter dataset," *Multimedia Tools and Applications*, pp. 1–18, 2021.

[84] D. K. Jain, A. Kumar, and A. Shrivastava, "Canardeep: a hybrid deep neural model with mixed fusion for rumour detection in social data streams," *Neural Computing and Applications*, pp. 1–12, 2022.

[85] S. A. Alkhodair, B. C. Fung, S. H. Ding, W. K. Cheung, and S.-C. Huang, "Detecting high-engaging breaking news rumours in social media," *ACM Transactions on Management Information Systems (TMIS)*, vol. 12, no. 1, pp. 1–16, 2020.

A Appendix

A.1 Guidelines for the annotators

Overview

Welcome to my annotation task. In this task we will present you with one tweet at a time and ask you to classify it. I will be using this data to build a machine that can recognise rumours based on certain features related to the tweet. You will be presented with around 250 tweets that have been gathered from Dutch Twitter, related one of three hashtags: #jinek, #vleestaks and #inflatie. These topics have been chosen because they seemed conducive to rumours and not for political reasons.

Concepts

We would like you to classify each tweet as being either a rumour or a non-rumour.

Classes For each tweet, please choose one of the following tags

Rumour: A rumour is a story or talking point that is (or could be) widely spread and is unverified. An example could be if someone said "Bob stole the cookies!", "I bet he needs them to build a fort!" or "He is coming for your cookies next!" What makes a rumour a rumour is that it is like a theory. One that could be easily shared around. Rumours often sound outrageous or wild. Though they do not always have to! **Non-Rumour:** This could either be a factual statement or an opinion. An example could be if someone said "There is video footage of someone that looks like Bob, sneaking into the kitchen at 3 am.", "I think Bob is just the type of person to do something like this.", or "What terrible news. Especially since Bob is on a diet..."

Note that rumours are not the same thing is false information. Wild, unconfirmed stories could still turn out to be true.

A.2 Plots

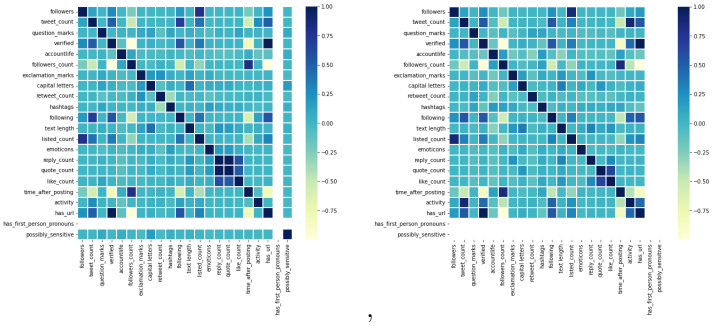

Fig. 9 Correlation matrix on tweets in *#jinek*)

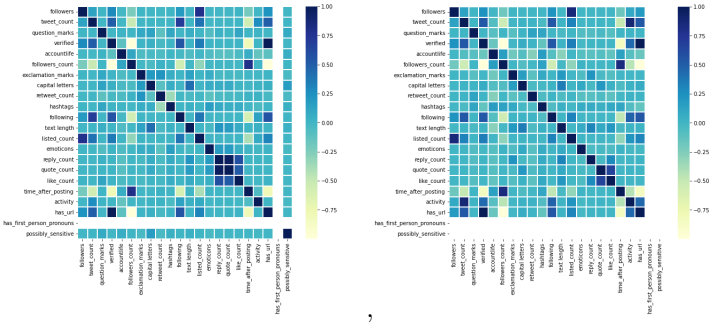

Fig. 10 Correlation matrix on tweets in *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	1000	11
	Rumour	500	130

Table 13 Confusion matrix for naive Bayes, using only "early" features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	990	13
	Rumour	510	130

Table 14 Confusion matrix for naive Bayes, using all features and all the hashtags, without self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	990	11
	Rumour	500	140

Table 15 Confusion matrix for naive Bayes, using only content features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	990	1
	Rumour	650	0

Table 16 Confusion matrix for naive Bayes, using only context features and all the hashtags, without self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	90	710
	Rumour	20	270

Table 17 Confusion matrix for naive Bayes, using all features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	24	3300
	Rumour	0	120

Table 18 Confusion matrix for naive Bayes, using all features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	890	0
	Rumour	2800	2

Table 19 Confusion matrix for naive Bayes, using all features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	380	19
	Rumour	140	6

Table 20 Confusion matrix for naive Bayes, using ony context features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	1600	0
	Rumour	60	0

Table 21 Confusion matrix for naive Bayes, using only context features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	440	0
	Rumour	1400	1

Table 22 Confusion matrix for naive Bayes, using only context features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	270	130
	Rumour	120	23

Table 23 Confusion matrix for naive Bayes, using content features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	1600	1
	Rumour	60	0

Table 24 Confusion matrix for naive Bayes, using content features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	440	0
	Rumour	1400	0

Table 25 Confusion matrix for naive Bayes, using content features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	1000
	Rumour	0	640

Table 26 Confusion matrix for linearSVM, using all features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	1000
	Rumour	0	640

Table 27 Confusion matrix for linearSVM, using only "early" features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	49	940
	Rumour	4	650

Table 28 Confusion matrix for linearSVM, using all features and all the hashtags, without self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	130	850
	Rumour	41	620

Table 29 Confusion matrix for linearSVM, using only content features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	5	490
	Rumour	0	320

Table 30 Confusion matrix for linearSVM, using only context features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	800
	Rumour	0	290

Table 31 Confusion matrix for linearSVM, using all features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	3300
	Rumour	0	120

Table 32 Confusion matrix for linearSVM, using all features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	230	650
	Rumour	860	2000

Table 33 Confusion matrix for linearSVM, using all features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	170	620
	Rumour	26	270

Table 34 Confusion matrix for linearSVM, using content features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	3300
	Rumour	0	120

Table 35 Confusion matrix for linearSVM, using content features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	890	0
	Rumour	2800	0

Table 36 Confusion matrix for linearSVM, using content features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	6	790
	Rumour	6	290

Table 37 Confusion matrix for linearSVM, using context features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	0	3300
	Rumour	0	120

Table 38 Confusion matrix for linearSVM, using context features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	36	850
	Rumour	110	2700

Table 39 Confusion matrix for linearSVM, using context features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	480	5
	Rumour	16	320

Table 40 Confusion matrix for XGB, using all features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	970	33
	Rumour	16	630

Table 41 Confusion matrix for XGBoost, using only content features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	130	500
	Rumour	76	330

Table 42 Confusion matrix for XGBoost, using all features in addition to the Bag of Words features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	940	8
	Rumour	22	670

Table 43 Confusion matrix for XGBoost, using only context features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	970	6
	Rumour	36	630

Table 44 Confusion matrix for naive Bayes, using only "early" features and all the hashtags, as well as self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	1000	2
	Rumour	4	630

Table 45 Confusion matrix for XGBoost, using all features and all the hashtags, without self-training.

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	240	160
	Rumour	120	27

Table 46 Confusion matrix for XGBoost, using all features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	20	1600
	Rumour	41	19

Table 47 Confusion matrix for XGBoost, using all features, split by hashtag so that the train set consists of *#jinek* and *#inflatie*, and the test set of *#vleestaks*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	390	57
	Rumour	1300	69

Table 48 Confusion matrix for XGBoost, using all features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	440	360
	Rumour	170	120

Table 49 Confusion matrix for XGBoost, using content features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	770	110
	Rumour	2100	730

Table 50 Confusion matrix for XGBoost, using content features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*

		Prediction label: Non-Rumour	Prediction label: Rumour
Actual Label	Non-Rumour	380	420
	Rumour	94	200

Table 51 Confusion matrix for XGBoost, using context features, split by hashtag so that the train set consists of *#vleestaks* and *#inflatie*, and the test set of *#jinek*

		Prediction label	
		Non-Rumour	**Rumour**
Actual Label	**Non-Rumour**	730	160
	Rumour	260	260

Table 52 Confusion matrix for XGBoost, using context features, split by hashtag so that the train set consists of *#jinek* and *#vleestaks*, and the test set of *#inflatie*